Searching for Ribbon-Shaped Paths in Fair Transition Systems

Marco Bozzano, Alessandro Cimatti, Stefano Tonetta, Viktoria Vozarova(✉)

Fondazione Bruno Kessler (FBK)
via Sommarive, 18
Trento 38123, Italy
{bozzano,cimatti,tonettas,vvozarova}@fbk.eu

Abstract. Diagnosability is a fundamental problem of partial observable systems in safety-critical design. Diagnosability verification checks if the observable part of system is sufficient to detect some faults. A counterexample to diagnosability may consist of infinitely many indistinguishable traces that differ in the occurrence of the fault. When the system under analysis is modeled as a Büchi automaton or finite-state Fair Transition System, this problem reduces to look for ribbon-shaped paths, i.e., fair paths with a loop in the middle.
In this paper, we propose to solve the problem by extending the liveness-to-safety approach to look for lasso-shaped paths. The algorithm can be applied to various diagnosability conditions in a uniform way by changing the conditions on the loops. We implemented and evaluated the approach on various diagnosability benchmarks.

Keywords: Diagnosability· Model checking · Liveness to safety

1 Introduction

The design of fault detection mechanisms is a standard part of the design of safety-critical systems. Faults are usually not directly observable. They are diagnosed by observing a sequence of observations and inferring the value of unobservable variables based on a system model. A fundamental question for the design of such partially observable systems is to determine if it always possible to detect a fault. Diagnosability verification is the problem of checking whether the available sensors are sufficient to determine the occurrence of a fault.

Historically, diagnosability verification is reduced to a model checking problem looking for a critical pair of indistinguishable traces that differ with respect to the fault. This pair witnesses the impossibility to detect the fault along such sequence of observations.

When considering fair transition systems, critical pairs are not sufficient and it is necessary to look for infinitely many indistinguishable traces. In case of finite state systems, such set of infinite traces can be represented by ribbon-shaped paths, i.e., paths with a loop in the middle. Previous solutions, hinted

D. Fisman and G. Rosu (Eds.): TACAS 2022, LNCS 13243, pp. 543–560, 2022.
https://doi.org/10.1007/978-3-030-99524-9_30

in [16], were based on either bounded model checking, so not able to prove diagnosability (absence of the critical ribbon-shaped paths) or BDD-based fixpoint computation, which suffers from the problem of precomputing the fair states.

In this paper, we propose a new approach based on the liveness-to-safety construction [3], where the search for a (single) lasso shaped path is reduced to an invariant property. Like in liveness-to-safety, we use additional variables to guess the loopback states, which in the case of ribbon-shaped paths are used twice, the first time for the loop in the middle, the second time for the final lasso. Additional constraints are added to encode the looping conditions that must hold in the two loops for encoding the diagnosability problem. The algorithm can be applied to various diagnosability conditions in a uniform way by changing the conditions on the loops. We implemented and evaluated the approach on various diagnosability benchmarks. Different algorithms have tested to solve the resulting invariant model checking problem, showing better performance with respect to the fixpoint-based approach.

The main contribution of the paper is the extension of liveness-to-safety to generate an infinite number of traces. The set is in the form of a ribbon shape (in other words, in the form $a; b^*; c; d^\omega$) and may have applications beyond the diagnosability problem, e.g., to solve non-interference problems requiring infinitely many different traces [13] or to counterexample-guided abstraction refinement.

The rest of the paper is organized as follows. In Section 2, we give an overview of related work. Section 3 defines the necessary formal background. The main problem along with the original solution is presented in Section 4. Our main contribution is introduced in Section 5, where we present the novel solution and prove its correctness. Section 6 contains the experimental evaluation comparing our solution with the original one. Finally, in Section 7 we give conclusions and directions for future work.

2 Related Work

The problem of diagnosability [17] refers to the possibility of inferring some desired information (e.g., the occurrence of a fault) during the execution of a system, in a partially observable environment. Hence, diagnosability can be phrased using hyperproperties, namely as a property of the traces representing the execution of the system [5,16].

In [16] it has been shown that the problem of diagnosability under fairness can be reduced to the search for ribbon-shaped paths, i.e. paths with a loop in the middle, where specific conditions on the occurrence of faults are imposed. Historically, diagnosability has been defined in the context of Discrete-Event Systems [17], without taking fairness into account. In [14] fairness is considered only in the context of live systems, i.e. under the hypothesis that every finite trace can be extended to an infinite fair trace, and fair diagnosability is introduced only informally. In this context, our ribbon-shaped fair critical pair corresponds to the critical pair of [14], where the faulty trace must be fair while the nominal trace may be unfair.

A construction similar to ribbon-shaped paths, called *doubly pumped lasso*, is used in [13] as a building block to address the problem of model checking a class of quantitative hyperproperties, as in the problem of quantitative non-interference (i.e., bound the amount of information about some secret inputs that may be leaked through the observable outputs of the system).

In [13] the problem of verifying quantitative hyperproperties is addressed using a model checking algorithm based on model counting, which is shown to have a better complexity than using an HyperLTL model checker, and a Max#SAT-based implementation. In [16], the authors address the problem of checking diagnosability using an extension of the classical twin-plant construction [15] and an LTL model checker. The approach we use in this paper builds upon the approach of [16], but uses an extension of the liveness-to-safety approach [3], instead. The extension omits the computation of fair states and keeps the representation of the system symbolic, which is more space efficient. The problem is reduced to the reachability problem. The problem is well-studied, thus we may take advantage of already developed algorithms for checking reachability.

3 Background

3.1 Symbolic Fair Transition Systems

The plant under analysis is represented as a finite-state *symbolic fair transition system* (SFTS). An SFTS is a tuple $\langle V, I, T, F\rangle$, where V is a finite set of Boolean state variables; I is a formula over V defining the initial states, T is a formula over V, V' (with V' being the next version of the state variables) defining the transition relation, and F is a set of formulas over V defining the fairness conditions. If $F = \emptyset$, we call it a *symbolic transition system* (STS) and write $\langle V, I, T\rangle$.

We remark that the choice of representing the plant in form of an SFTS does not restrict the generality of the framework. In fact, it is possible to encode labeled transition systems and discrete event systems.

A *state* s is an assignment to the state variables V. We denote with s' the corresponding assignment to V'. Given an assignment to a set V of Boolean variables, we also represent the assignment as the set of variables that are assigned to true. Given a state s and a subset of variables U, we denote with $s_{|U}$ the restriction of s to the variables in U.

In the following we assume that an SFTS $P \doteq \langle V, I, T, F\rangle$ is given.

Given a sequence of states σ, we denote with σ^k the sequence obtained by repeating σ for k times, and σ^ω the sequence obtained by repeating σ for an infinite number of times.

Given a state s_0 of P, a *trace of P starting from s_0* is an infinite sequence $\pi \doteq s_0, s_1, s_2, \ldots$ of states starting from s_0 such that, for each $k \geq 0$, $\langle s_k, s_{k+1}\rangle$ satisfies T, and for all $f \in F$, for infinitely many $i \geq 0$, the formula f is true in

s_i. If s_0 is initial, i.e., it satisfies I, then we say that π is a *trace of* P. We write Π_P for the set of traces of P.

We denote with $\pi[k]$ the $k+1$-th state s_k of π. We say that s is *reachable* (in k steps) in P iff there exists a sequence $\pi = s_0 s_1 \ldots s_k$, where $s_k = s$, s_0 satisfies I and every $\langle s_i, s_{i+1}\rangle$ satisfies T. A state s is fair if there exists a trace starting from s.

Given a trace $\pi \doteq s_0, s_1, s_2, \ldots$ and a subset of variables $U \subseteq V$, we denote by $\pi_{|U} \doteq s_{0|U}, s_{1|U}, s_{2|U}, \ldots$ the projection over the variables in U.

Let $S^1 = \langle V^1, I^1, T^1, F_1\rangle$ and $S^2 = \langle V^2, I^2, T^2, F_2\rangle$ be two SFTSs. We define a *synchronous product* $S^1 \times S^2$ as the SFTS $\langle V^1 \cup V^2, I^1 \wedge I^2, T^1 \wedge T^2, F_1 \cup F_2\rangle$. Every state s of $S^1 \times S^2$ is an assignment to the two sets of state variables V^1 and V^2 such that $s^1 = s_{|V^1}$ is a state of S^1 and $s^2 = s_{|V^2}$ is a state of S^2.

Let p be a propositional formula over V. We write $s \models p$ iff s satisfies p, and $\pi, i \models p$ if $\pi[i]$ satisfies p. We write $P \models p$ iff for all reachable s in P it holds that $s \models p$. Let φ be a formula over an infinite trace expressed in LTL [12]. We write $\pi \models \varphi$ iff φ is true on the trace π. We write $P \models \varphi$ iff for all traces π in Π_P it holds that $\pi \models \varphi$.

In the rest of the presentation, we sometimes use a context, which we express as an LTL formula Ψ, to restrict the set of traces of the plant. This is useful to address the problem of diagnosability under assumptions. Note that, since our framework supports plants with fairness constraints, the incorporation of the context can be done (see, e.g., [10]) by converting the context into an SFTS S_Ψ (representing the monitor automaton for the LTL formula) and replacing the plant P with $P \times S_\Psi$ (the synchronous product of the plant with the monitor automaton).

The Twin Plant Construction The twin plant construction of a plant P over a subset $Y \subseteq V$ of variables (the observable variables), denoted $\textsc{Twin}(P, Y)$ and originally proposed by [15], is based on two copies of P, such that a trace in the twin plant corresponds to a pair of traces of P. In the security domain, two copies of a system used for verification are known as a self-composition [2].

The twin plant can be defined as the synchronous product of two copies of the SFTS corresponding to the plant. Formally, given a plant $P = \langle V, I, T, F\rangle$, we denote with $P_L \doteq \langle V_L, I_L, T_L, F_L\rangle$ and $P_R \doteq \langle V_R, I_R, T_R, F_R\rangle$ the ('left' and 'right') copies of P, obtained by renaming each variable v as v_L or v_R, respectively (i.e., if $\Box \in \{L, R\}$, then $V_\Box$ stands for the set of variables $\{v_\Box \mid v \in V\}$. Moreover, we define a formula $\textsc{ObsEq}$ stating that the sets of observable variables of the two copies are equal at the given point. The twin plant of P is defined as follows.

Definition 1 (Twin Plant). *Given a set of variables* $Y \subseteq V$, *the* twin plant *of* $P = \langle V, I, T, F\rangle$ *is the SFTS* $\textsc{Twin}(P, Y) \doteq P_L \times P_R$. *Moreover, we define the formula* $\textsc{ObsEq} \doteq \bigwedge_{v \in Y} v_L = v_R$.

There is a one-to-one correspondence between $\Pi_P \times \Pi_P$ (pairs of traces of P) and $\Pi_{\textsc{Twin}(P,Y)}$ (traces of $\textsc{Twin}(P, Y)$). A trace of $\textsc{Twin}(P, Y)$: $\pi \doteq (s_{0,L}, s_{0,R})$,

$(s_{1,L}, s_{1,R}), \ldots$ can be decomposed into two traces of P: $\mathit{Left}(\pi) \doteq s_{0,L}, s_{1,L}, \ldots$ and $\mathit{Right}(\pi) \doteq s_{0,R}, s_{1,R}, \ldots$. Conversely, given two traces π_L and π_R in Π_P, there is a corresponding trace in $\Pi_{\textsc{Twin}(P,Y)}$, denoted by $\pi_L \times \pi_R$.

3.2 Liveness to Safety (L2S).

The *liveness-to-safety reduction* (L2S) [3] is a technique for reducing an LTL model checking problem on a finite-state transition system to an invariant model checking problem. The idea is to encode the absence of a lasso-shaped path violating the LTL property $\mathbf{FG}\neg f$ as an invariant property.

The encoding is achieved by transforming the original transition system S to the transition system S_{L2S}, introducing a set $\overline{X}$ of variables containing a copy $\overline{x}$ for each state variable x of the original system, plus additional variables *seen*, *triggered* and *loop*. Let $S \doteq \langle X, I, T\rangle$. L2S transforms the transition system in $S_{\text{L2S}} \doteq \langle X_{\text{L2S}}, I_{\text{L2S}}, T_{\text{L2S}}\rangle$ so that $S \models \mathbf{FG}\neg f$ if and only if $S_{\text{L2S}} \models \neg \mathit{bad}_{\text{L2S}}$, where:

$$\begin{aligned}
X_{\text{L2S}} &\doteq X \cup \overline{X} \cup \{\mathit{seen}, \mathit{triggered}, \mathit{loop}\}\\
I_{\text{L2S}} &\doteq I \wedge \neg \mathit{seen} \wedge \neg \mathit{triggered} \wedge \neg \mathit{loop}\\
T_{\text{L2S}} &\doteq T \wedge \left[\textstyle\bigwedge_X \overline{x} \iff \overline{x}'\right]\\
&\quad \wedge\left[\mathit{seen}' \iff (\mathit{seen} \vee \textstyle\bigwedge_X (x \iff \overline{x}))\right]\\
&\quad \wedge\left[\mathit{triggered}' \iff (\mathit{triggered} \vee (f \wedge \mathit{seen}'))\right]\\
&\quad \wedge\left[\mathit{loop}' \iff (\mathit{triggered}' \wedge \textstyle\bigwedge_X (x' \iff \overline{x}'))\right]\\
\mathit{bad}_{\text{L2S}} &\doteq \mathit{loop}
\end{aligned}$$

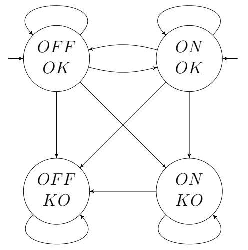
The variables $\overline{X}$ are used to non-deterministically guess a state of the system from which a reachable fair loop starts. The additional variables are used to remember that the guessed state was seen once and that the signal f was true at least once afterwards.

4 The Problem of Ribbon-Shaped Paths

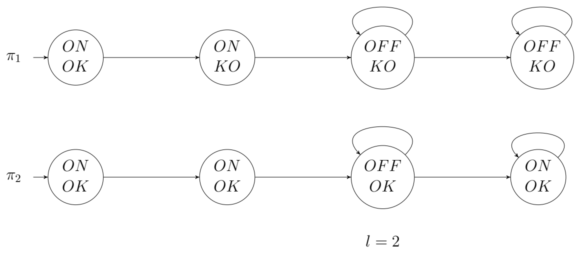
4.1 The Diagnosability Problem

The *observable part* $\mathit{obs}(s)$ of a state s is the projection of s on the subset Y of observable state variables. Thus, $\mathit{obs}(s) \doteq s_{|Y}$. The observable part of π is $\mathit{obs}(\pi) \doteq \mathit{obs}(s_0), \mathit{obs}(s_1), \mathit{obs}(s_2), \ldots = \pi_{|Y}$. Given two traces π_1 and π_2, we denote by $\textsc{ObsEqUpTo}(\pi_1, \pi_2, k)$ the condition saying that, for all i, $0 \leq i \leq k$, $\mathit{obs}(\pi_1[i]) = \mathit{obs}(\pi_2[i])$.

Let β be a formula over V representing the fault condition to be diagnosed. We call β a diagnosis condition. A system is diagnosable for β if there exists a bound d such that after the occurrence of β, an observer can infer within d steps that β indeed occurred. This means that any other trace with the same observable part contains β as well. Formally, it was first defined in [17] as follows.



(a) light bulb



(b) RCP

Fig. 1: The light bulb example and an example of a ribbon-shaped critical pair in the light bulb.

Definition 2 (Diagnosability). *Let P be a plant and β a diagnosis condition. P is* diagnosable *for β iff there exists $d \geq 0$ such that for every trace π_1 and index $i \geq 0$ such that $\pi_1, i \models \beta$, it holds:*

$$(\exists j \in \mathbb{N}\ i \leq j \leq i+d \cdot (\forall \pi_2 \cdot \textsc{ObsEqUpTo}(\pi_1, \pi_2, j) \Rightarrow \exists k \in \mathbb{N}\ k \leq j \cdot \pi_2, k \models \beta)).$$

The above definition requires a global bound, while when considering fair transition systems it is possible that the occurrence of β can be inferred eventually, but without a fixed bound. That is the motivation of extending the definition to fair diagnosability [16].

Definition 3 (Fair Diagnosability). *Let P be a plant and β a diagnosis condition. P is* fair-diagnosable *for β iff for every trace π_1, there exists $d \geq 0$ such that for every index $i \geq 0$ such that $\pi_1, i \models \beta$, it holds:*

$$(\exists j \in \mathbb{N}\ i \leq j \leq i+d \cdot (\forall \pi_2 \cdot \textsc{ObsEqUpTo}(\pi_1, \pi_2, j) \Rightarrow \exists k \in \mathbb{N}\ k \leq j \cdot \pi_2, k \models \beta)).$$

Example 1. Consider the state machine of a light bulb as shown in Figure 1a, with the observable value OFF/ON and the diagnosis condition $\beta \doteq KO$. Consider the following context: $\mathsf{G}(KO \rightarrow \mathsf{F}\, OFF) \wedge \mathsf{G}(OK \rightarrow \mathsf{F}\, ON)$. Intuitively, the LTL formula states that globally a state where KO holds is followed eventually by a state where OFF holds, and similarly a state where OK holds is followed eventually by a state where ON holds. Therefore, if the execution reaches KO, it will eventually go into state OFF/KO and remain there forever. If an execution is instead always OK, then it will visit infinitely often the state ON/OK. We can prove that condition β is not fair-diagnosable according to Def. 3. In fact, for every j, there exists a trace without β that is observationally equivalent up to j to the trace with β. Notice how the fairness condition causes the observations after a failure to always diverge *eventually*, but that this event can be delayed indefinitely.

4.2 Ribbon-Shaped Critical Pairs

Figure 2 illustrates the concept of ribbon-shaped paths. The formal definition is as follows.

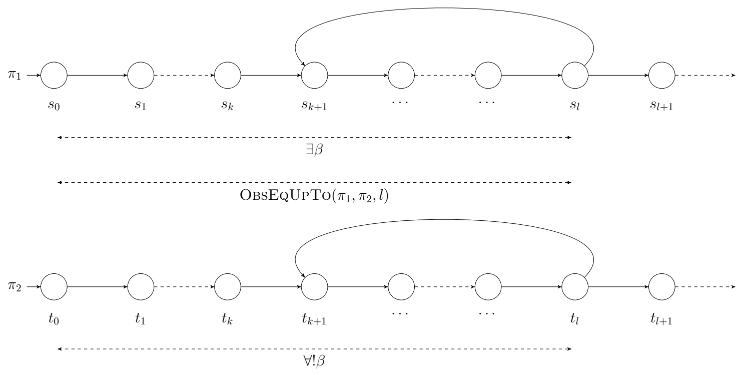


Fig. 2: Ribbon-shaped critical pair

Definition 4 (Ribbon-Shaped Critical Pairs (RCP)). *Let P be a plant and β a diagnosis condition. We say that $\pi_1, \pi_2 \in \Pi_P$ are a ribbon-shaped critical pair for the diagnosability of β iff there exist k, l such that $0 \leq k \leq l$ and:*

1. $\pi_1[l] = \pi_1[k]$ *and* $\pi_2[l] = \pi_2[k]$*;*
2. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, l)$*;*
3. $\pi_1, i \models \beta$ *for some* $i, 0 \leq i \leq l$*;*
4. $\pi_2, i \not\models \beta$ *for all* $i, 0 \leq i \leq l$*.*

For fair diagnosability, the definition is similar:

Definition 5 (Ribbon-Shaped Fair Critical Pairs (RFCP)). *Let P be a plant and β a diagnosis condition. We say that $\pi_1, \pi_2 \in \Pi_P$ are a ribbon-shaped fair critical pair for the diagnosability of β iff there exist k, l such that $0 \leq k \leq l$ and:*

1. $\pi_1[l] = \pi_1[k]$ *and* $\pi_2[l] = \pi_2[k]$*;*
2. π_1 *is in the form* $s_0, s_1, \dots s_k, (s_{k+1}, \dots s_l)^\omega$*;*
3. $\textsc{ObsEqUpTo}(\pi_1, \pi_2, l)$*;*
4. $\pi_1, i \models \beta$ *for some* $i \geq 0$*;*
5. $\pi_2, i \not\models \beta$ *for all* $i, 0 \leq i \leq l$*.*

In this paper, we use a slightly different definition than the one given in [16]. Definition 5 includes an additional constraint on π_1 by requiring a loop shape. However, these two definitions are equivalent and the proof of it can be found in the extended version of the paper.

Example 2. Fig. 1b shows an example of a ribbon-shaped critical pair for the light bulb of Example 1.

We can prove that, in the general case, β is not diagnosable if and only if there exists a ribbon-shaped critical pair. In other words, ribbon-shaped critical pairs are necessary and sufficient for diagnosability violation. The following theorem is adapted and extended from [16] and can be proved in a similar way.

Theorem 1 (RCP necessary and sufficient for diagnosability). *Let P be a plant. P is not diagnosable for β iff there exists a ribbon-shaped critical pair for the diagnosability of β. P is not fair diagnosable for β iff there exists a ribbon-shaped fair critical pair for the fair diagnosability of β.*

The proof can be done similarly as in [16]. The theorem in [16] is proved for asynchronous systems while here we assume that the plants in the twin plant are synchronized on the observable part.

4.3 Fixpoint-based Algorithm

The ribbon-shaped structure requires to eventually reach a loop (the ribbon), from which it is possible to branch with a fair suffix (the final lasso). Therefore, it combines path and branching conditions, and can be encoded into a CTL* formula [16] over variables of the twin plant. We can verify whether the formula holds in the twin plant using a fixpoint-based algorithm. Actually, the specific structure allows for a simple implementation on top of standard BDD-based model checking [16]: it is sufficient first to compute the set of fair states, then to compute the set of fair states staying forever in the looping condition, and finally to look for an initial state reaching such loop.

The main issue of this approach is the computation of fair states, which is performed independently from the diagnosis condition and may be a bottleneck in case of complex fairness conditions.

5 Extended Liveness to Safety

In this section, we propose a novel algorithm for finding RCPs and RFCPs in fair symbolic transition systems. The algorithm extends L2S such that it searches for two consequent loops instead of one. We define a ribbon structure, which is constructed from the twin plant. The ribbon structure is parametrized, thus it can be used for finding both RCPs and RFCPs with only a slight modification. We prove that a certain state is reachable in the ribbon structure if and only if there exists an RCP/RFCP in the original structure.

The ribbon structure extends the twin plant of the original structure with a new copy of state variables, new flags, and new transitions that constrain the behaviour of the new variables and the flags. In the following, we describe how the twin plant is extended and we formally define the ribbon structure.

5.1 Definition of the L2S Extension

The ribbon structure is parametrized by SFTS P, two propositional formulas p and q and two sets of propositional fairness conditions F_1 and F_2. These parameters are later instantiated depending on the specific ribbon-shaped path that is considered. In particular, p represents the diagnosis condition β in the left copy of the twin plant and q represents the negation of β in the right copy conjoined with the constraint to force the same observations on the two copies.

The ribbon structure $P_\sim$ and the propositional formula $\varphi_\sim$ are defined such that any path ρ of $P_\sim$ on which $\varphi_\sim$ is reached satisfies the following conditions:

- ρ contains two consequent loops L_1 and L_2 that satisfy fairness conditions F_1 and F_2 respectively;
- p is satisfied in some state of ρ before the end of the first loop;
- q is satisfied in all states of ρ before the end of the first loop.

In the rest of this section, we formally define the set of variables, the initial formula and the transition formula using the parameters described above.

Variables Similarly as in the original liveness-to-safety reduction, we create a copy of all state variables of the twin plant. The copy variables serve as a guess of the state representing a loopback. The variables are denoted by overline and defined as $\overline{V} = \{\overline{v} \mid v \in V\}$, where V is the set of variables of the twin plant P. The variables are reused both for the first and the second loop, where the second loop is a fair loop.

The flags are auxiliary variables used to monitor whether a loop was found and whether all loop conditions were satisfied. The set of flags is defined as $V_\mathbf{m} = \{\mathbf{m}_{seen}, \mathbf{m}_{L_1}, \mathbf{m}_p, \mathbf{m}_q\} \cup \bigcup_{f_i \in F_1} \mathbf{m}_{1,i} \cup \bigcup_{f_i \in F_2} \mathbf{m}_{2,i}$. The intuition behind each flag is as follows:

$\mathbf{m}_{seen}$ is true	$\Longleftrightarrow$	the loopback (either the first or the second one) was already seen and is saved in $\overline{V}$;
$\mathbf{m}_{L_1}$ is true	$\Longleftrightarrow$	the first loop was already found;
$\mathbf{m}_p$ is true	$\Longleftrightarrow$	p was true;
$\mathbf{m}_q$ is true	$\Longleftrightarrow$	q was true in all previous states;
$\mathbf{m}_{1,i}$ is true	$\Longleftrightarrow$	$f_i \in F_1$ was true in the first loop;
$\mathbf{m}_{2,i}$ is true	$\Longleftrightarrow$	$f_i \in F_2$ was true in the second loop.

In addition, when $\mathbf{m}_{seen}$ is true, the current state is in a loop. If $\mathbf{m}_{L_1}$ is false, it is in the first loop. Otherwise, the first loop was already found and the current state is in the second loop.

Auxiliary Formula The following formula φ_{L_1} states requirements for finding the loopback of the first loop L_1. We need that the conditions on p and q are satisfied and that L_1 was yet not found. In addition, we need that the fairness conditions were true and that the current state is the same as the guessed loopback.

$$\varphi_{L_1} := \mathbf{m}_p \wedge \mathbf{m}_q \wedge \neg\mathbf{m}_{L_1} \wedge \mathbf{m}_{seen} \wedge \bigwedge_{f_i \in F_1} \mathbf{m}_{1,i} \wedge \bigwedge_{v \in V} v = \overline{v}$$

Initial Formula All flags besides $\mathbf{m}_q$ are initialized to false, $\mathbf{m}_q$ is initialized to true:

$$\neg\mathbf{m}_{seen} \wedge \neg\mathbf{m}_{L_1} \wedge \neg\mathbf{m}_p \wedge \mathbf{m}_q \wedge \bigwedge_{f_i \in F_1} \neg\mathbf{m}_{1,i} \wedge \bigwedge_{f_i \in F_2} \neg\mathbf{m}_{2,i} \qquad \text{(I1)}$$

Transition Formulas We define transitions (T1)–(T8) to ensure the correct behaviour of the introduced variables such that the conditions mentioned above are satisfied. The transitions and their intuitive descriptions are as follows.

– Anytime $\mathbf{m}_{seen}$ is set to true, in the next state the copied variables are set to the state variables of the current state:

$$\neg\mathbf{m}_{seen} \wedge \mathbf{m}_{seen}{}' \implies \bigwedge_{v\in V} \overline{v}' = v \tag{T1}$$

– If $\mathbf{m}_{seen}$ is true, the values of the copy variables are preserved also in the next state:

$$\mathbf{m}_{seen} \implies \bigwedge_{v\in V} \overline{v}' = \overline{v} \tag{T2}$$

– The flags $\mathbf{m}_{x,i}$ can change to true only when $f_i \in F_x$ is true and the current state is in L_x:

$$\bigwedge_{f_i\in F_1} \big((\mathbf{m}_{1,i}{}' = \mathbf{m}_{1,i}) \vee (\mathbf{m}_{i,1}{}' \wedge \mathbf{m}_{seen} \wedge \neg\mathbf{m}_{L_1} \wedge f_i) \big) \tag{T3}$$

$$\bigwedge_{f_i\in F_2} \big((\mathbf{m}_{2,i}{}' = \mathbf{m}_{2,i}) \vee (\mathbf{m}_{2,i}{}' \wedge \mathbf{m}_{seen} \wedge \mathbf{m}_{L_1} \wedge f_i) \big) \tag{T4}$$

– $\mathbf{m}_{L_1}$ can change to true only when the first loop was found, as specified above by φ_{L_1}, and it forces $\mathbf{m}_{seen}$ to be set to false:

$$(\mathbf{m}_{L_1}{}' = \mathbf{m}_{L_1}) \quad \vee \quad (\varphi_{L_1} \wedge \neg\mathbf{m}_{seen}{}' \wedge \mathbf{m}_{L_1}{}') \tag{T5}$$

– $\mathbf{m}_{seen}$ can change to false only when L_1 was just found:

$$\mathbf{m}_{seen} \implies (\mathbf{m}_{seen}{}' \quad \vee \quad (\neg\mathbf{m}_{seen}{}' \wedge \neg\mathbf{m}_{L_1} \wedge \mathbf{m}_{L_1}{}')) \tag{T6}$$

– $\mathbf{m}_p$ can change to true only when p is true:

$$(\mathbf{m}_p{}' = \mathbf{m}_p) \vee (p \wedge \mathbf{m}_p{}') \tag{T7}$$

– Anytime q is false, $\mathbf{m}_q$ goes to false and stays false:

$$(\neg\mathbf{m}_q \vee \neg q) \implies \neg\mathbf{m}_q{}' \tag{T8}$$

Note that the transitions (T3), (T4), (T5) and (T7) imply that flags $\mathbf{m}_{x,i}$, $\mathbf{m}_{L_1}$, $\mathbf{m}_p$ can change their value from false to true only once and then they stay true. The transition (T8) implies that $\mathbf{m}_q$ can change its value from true to false only once and then it stays false. Finally, (T6) implies that $\mathbf{m}_{seen}$ is set to false exactly once, when L_1 is found, and thus set to true exactly twice, when a loopback of either L_1 or L_2 is guessed.

Ribbon Structure Putting together the variables and formulas defined above, we give the following definition of the ribbon structure.

Definition 6. *For the plant $P = \langle V, I, T, F\rangle$, the propositional formulas p, q and the sets of propositional formulas F_1, F_2 over V, let $\langle V_\sim, I_\sim, T_\sim\rangle$ be a symbolic transition system where:*

- $V_\sim = V \cup \overline{V} \cup V_{\mathbf{m}}$;
- $I_\sim = I \wedge (I1)$;
- $T_\sim = T \wedge (T1) \wedge (T2) \wedge (T3) \wedge (T4) \wedge (T5) \wedge (T6) \wedge (T7) \wedge (T8)$.

We call this STS a ribbon structure and denote it by $\textsc{Ribbon}(P, p, q, F_1, F_2)$.

To finish the reduction, we define the reachability condition. Intuitively, the condition should express that the second loop was found. This means that the first loop was already found, all fairness conditions in F_2 were true and the current state is the same as the guessed loopback:

$$\varphi_\sim := \mathbf{m}_{L_1} \wedge \mathbf{m}_{seen} \wedge \bigwedge_{f_i \in F_2} \mathbf{m}_{2,i} \wedge \bigwedge_{v \in V} v = \overline{v}.$$

In the next section, we show how the reachability in a ribbon structure is used to find RCPs and RFCPs and we prove that our construction is correct.

5.2 Correctness

The ribbon structure and the reachability condition are defined such that any satisfiable trace contains two consecutive loops. The definitions of RCP and RFCP describe only the first loop. Not all critical pairs contain the second loop. However, using the following propositions, we claim that the existence of a critical pair implies existence of a critical pair with two loops, where the first loop is as in the original pair and the second loop is fair. This fact is necessary to prove that if P contains a critical pair, we can find a critical pair with two loops in the ribbon structure.

Proposition 1. *Let π be a trace of an SFTS P. Then, any prefix of π can be extended to a trace π_F that ends with a fair loop.*

Proposition 2. *Let $\pi_1 = s_1, s_2, s_3 \ldots$, $\pi_2 = t_1, t_2, t_3 \ldots$ be traces of SFTS P that end with a fair loop. Then, the path $(s_1, t_1), (s_2, t_2), (s_3, t_3) \ldots$ is a trace of $P \times P$ that ends with a fair loop.*

The first proposition is true because we consider only finite systems. In a finite system, any infinite fair suffix contains a state that is repeated infinitely many times. Thus, there must be two occurrences of the state in between which all fairness conditions are true at least once. The second proposition is true because we can unroll the fair loops of π_1 and π_2 until both of them are in loop and then we match the period of the new fair loop in $\pi_1 \times \pi_2$ by taking the least common multiple of periods of the fair loops.

Theorem 2. *Let P be a plant and $P_\sim = \textsc{Ribbon}(\textsc{Twin}(P, Y), p, q, F_1, F_2)$ is a ribbon structure where $p = \beta_L$, $q = \neg\beta_R \wedge \textsc{ObsEq}$, $F_1 = \emptyset$, $F_2 = F_L \cup F_R$. There exists a ribbon-shaped critical pair in P for the diagnosability of β iff $P_\sim \models \varphi_\sim$.*

Proof. Here, we sketch the proof of the theorem. The full proof is given in the extended version of the paper. We separately prove both directions of the equivalence:

$\Longrightarrow$ We have $\pi_1, \pi_2 \in \Pi_P$ satisfying Definition 4. We prove that there is a trace $\rho \in \Pi_{P_\sim}$ such that $\rho \models \varphi_\sim$. At first, we show what the trace looks like and then we prove it is a trace of $P_\sim$. Let $\pi_{1,F}$, $\pi_{2,F}$ be a critical pair with two loops, where the first loop is equal to the loop in π_1, π_2 and the second loop is fair. We construct the path ρ as symbolized in Figure 3. The main idea is to set $\rho_{|V}$ to $\pi_{1,F} \times \pi_{2,F}$. The existence of loop bounds k, l, k', l' follows from the definition of RCP. In the copy variables $\rho_{|\overline{V}}$, we keep $(\pi_{1,F} \times \pi_{2,F})[k]$ until the first loop is found and then we switch to $(\pi_{1,F} \times \pi_{2,F})[k']$. Flags $\mathbf{m}_{seen}$ and $\mathbf{m}_{L_1}$ are set accordingly to the bounds of the loops. Flags $\mathbf{m}_p$ and $\mathbf{m}_{2,i}$ are set to true after conditions β_L and f_i respectively were true. The existence of such states where the conditions are satisfied follows from the definition of RCP. Flag $\mathbf{m}_q$ is true until the first loop is found, because from the definition of RCP we know that $\neg\beta_R$ and $\textsc{ObsEq}$ are true.
The formal definition of ρ and the full proof that $\rho \in \Pi_{P_\sim}$ and $\rho \models \varphi_\sim$ is given in the appendix.

$\Longleftarrow$ We have $P_\sim \models \varphi_\sim$, thus there is $\rho \in \Pi_{P_\sim}$ such that $\rho \models \varphi_\sim$. Assume we have such ρ. We show how to construct π_1 and π_2 from ρ and then we prove that π_1, π_2 are an RCP for P and β. Let us set $\pi_1 = \rho_{|V_L}$ and $\pi_2 = \rho_{|V_R}$. Let the bounds k, l, k', l' of the loops in π_1 and π_2 be the indices:
- l' is such that $\rho, l' \models \varphi_{L_2}$;
- $k' < l'$ is the greatest index such that $\rho, k' \models \neg\mathbf{m}_{seen}$;
- $l < k' + 1$ such that $\rho, l \models \neg\mathbf{m}_{L_1} \wedge \mathbf{m}_{L_1}{}'$, from the construction we know there is only one such l;
- $k < l$ is the greatest index such that $\rho, k \models \neg\mathbf{m}_{seen}$.

In Appendix A, we finish the prove by showing that π_1 and π_2 are an RCP.

Theorem 3. *Let P be a plant and $P_\sim = \textsc{Ribbon}(\textsc{Twin}(P, Y), p, q, F_1, F_2)$ is a ribbon structure where $p = \beta_L$, $q = \neg\beta_R \wedge \textsc{ObsEq}$, $F_1 = F_L$, $F_2 = F_L \cup F_R$. There exists a ribbon-shaped critical pair in P for the fair diagnosability of β iff $P_\sim \models \varphi_\sim$.*

The proof is very similar to the previous one. The only difference is the necessity to verify the fairness of the first loop, which is done the same way as the fairness of the second loop and thus straightforward.

6 Experimental Evaluation

We compared the proposed technique based on L2S and the technique based on the computation of fixpoints using BDD proposed in [16] and briefly described

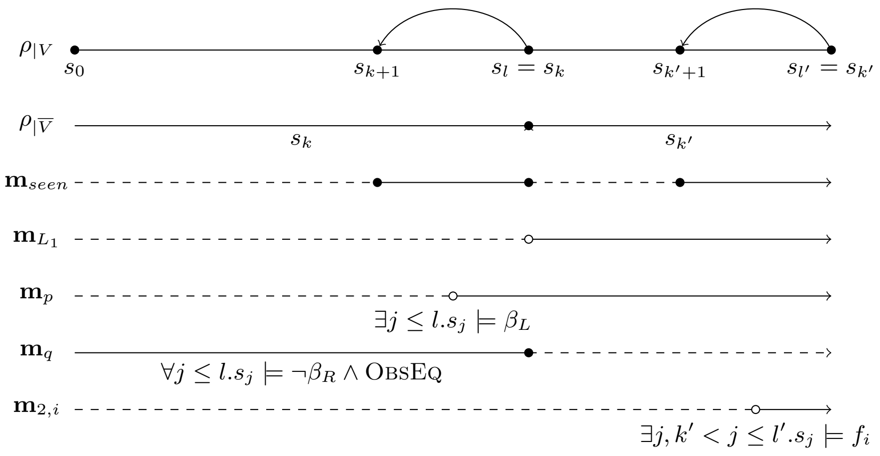


Fig. 3: The trace ρ as constructed in proof of Theorem 2. For each $\mathbf{m} \in V_{\mathbf{m}}$, a dashed line means $\rho, i \models \neg\mathbf{m}$, a full line and a full circle mean $\rho, i \models \mathbf{m}$, an empty circle means $\rho, i+1 \models \mathbf{m}$.

in Section 4.3. We implemented both algorithms in the xSAP platform [4] and tested them on benchmarks. The benchmarks, the tool and the scripts required to test it can be found online[1]. In this section, we at first introduce the implementation of the proposed technique and we describe the benchmarks. Then, we show comparison of the two techniques and we comment on their performance.

6.1 Implementation

We have implemented both the L2S algorithm and the BDD-based algorithm inside of the xSAP tool [4]. The algorithms make use of various procedures already implemented in nuXmv [8] and integrated in xSAP, mainly computation of fixpoint with BDDs [7] and different invariant model checking algorithms. The fair states are computed with the Emerson-Lei doubly-fixpoint algorithm [1]. The invariant model checking is implemented using engines based on standard verification algorithms IC3 [6], k-induction [18] and BDD-based fixpoint [11].

The input of each algorithm is a model in an SMV language[2], a list of observable variables of the model, a propositional diagnosis condition and an LTL formula representing the context. Both the model and the context are translated into Büchi automata and their parallel composition with the union of their accepting states is computed. The resulting set of accepting states is the set of fairness conditions. Then, a twin plant is constructed. The fixpoint-based algorithm is described in Section 4.

[1] http://es.fbk.eu/people/vvozarova/diag-rcp-search.zip
[2] see nuXmv manual htpps://nuxmv.fbk.eu)

Table 1: Properties of the used models.

model	#bool var	#reach	diam	#obs	#fairness
acex	31	$2^{19.4}$	96	5-21	1
autogen	99	$2^{12.0}$	20	4-20	1-4
cassini	176	$2^{44.2}$	8	5-58	1
guidance	98	$2^{47.5}$	70	5-62	1
pdist	83	$2^{11.0}$	31	5-41	1-4

In the L2S algorithm, we get the ribbon structure $P_{\sim}$ and the propositional formula $\varphi_{\sim}$ by extending the twin plant with new variables and transitions as defined in Section 6. The parameters p and q are constructed from the diagnosis condition and the set of observable variables. Finally, an arbitrary reachability algorithm is used to solve the reachability of $\varphi_{\sim}$ in the resulting system.

6.2 Benchmarks

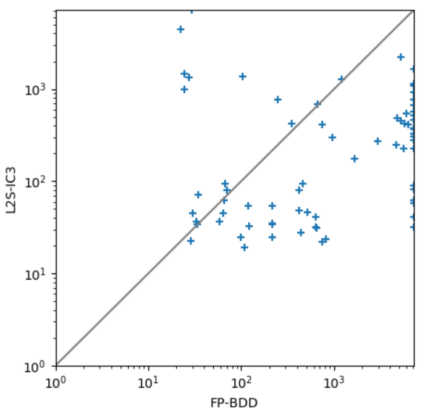
We selected several benchmarks modelling industrial use cases. The models are finite with boolean variables. For each model, we have specified a fault condition and possibly more sets of fairness conditions. Both the fault condition and the fairness conditions are given as propositional formulas. In Table 1, we give for each model the number of variables, the number of reachable states, the diameter of the state space, the sizes of sets of observable variables and the sizes of fairness condition sets.

Each benchmark was tested with more sets of observable variables and some were tested with more sets of fairness conditions. In sum, we have 72 examples for diagnosability and fair diagnosability problems and each instance was solved by BDD-based fixpoint approach and L2S approach with IC3, k-induction and BDD engines. This gives the total of 576 individual invocations of the xSAP tool. The experiments were run in parallel on a cluster with nodes with Intel Xeon CPU running at 2.27GHz with 8CPU, 48GB. The timeout for each run was two hours and the memory cap was set to 8GB.

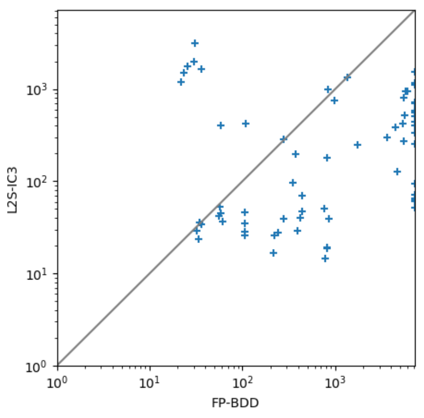
6.3 Results

The results for selected examples are given in Table 2 and all results are plotted in Figure 4a for diagnosability and in Figure 4b for fair diagnosability. We compare the BDD-based fixpoint algorithm (FP-BDD) with L2S with IC3 engine algorithm (L2S-IC3). The k-induction engine was unable to prove diagnosability with the given bound on k (150), time and memory. In general, it performs better on cases where a counterexample exists, which are not of concern in this paper. The runs for L2S with BDD engine reached timeout in 127 out of 144 cases and for this reason we do not include it in the analysis.

As both figures and the table show, the approach using L2S extension is in most cases more effective than the BDD-based approach proposed in the previous



(a) diagnosability



(b) fair diagnosability

Fig. 4: Results for the diagnosability (a) and the fair diagnosability (b) comparing L2S approach with IC3 engine and BDD-based fixpoint computation approach. The axes represent time in seconds on a logarithmic scale.

literature. The novel technique manages to outperform the previous one in most cases, as is shown by the cases plotted below the diagonal line on each figure. Moreover, it manages to solve some cases in which the fixpoint-based algorithm timed out. For the acex model, FP-BDD performs better than L2S-IC3. This is because the model has few boolean variables, thus BDDs are smaller and operations on them are faster. In addition, IC3 needs 56-116 frames to prove non-reachability on acex, compared to 3-62 frames in other cases.

7 Conclusions and Future Work

In this paper, we considered the problem of proving the absence of a ribbon-shaped path, which is a core issue in proving diagnosability of fair transition systems. We conceived a new encoding extending the liveness-to-safety paradigm in order to search for two consecutive loops. We implemented the algorithm in the xSAP tool and evaluated it on various diagnosability benchmarks in comparison with a fixpoint-based solution.

The directions for future work are manifold: first, generalize the looping conditions to consider also problems different from diagnosability such as non-interference properties (as in [13]); second, exploit the generation of infinite sets of traces in counterexample-guided abstraction refinement, reducing the number of refinement iterations; finally, extend the approach to infinite-state systems, taking into account data variables that are updated in the loop (as in [9]).

Table 2: Results comparing L2S with IC3 engine and BDD-based algorithm. The times are given in seconds, TO stands for the timeout of 7200 seconds. All cases are diagnosable.

model	#obs	#fairness	diagnosability		fair diagnosability	
			L2S-IC3	FP-BDD	L2S-IC3	FP-BDD
acex	5	1	TO	**29.59**	3114.25	**30.58**
	9	1	4385.18	**22.25**	1493.26	**23.22**
	13	1	992.60	**24.25**	1203.87	**21.94**
	17	1	1450.14	**24.65**	1754.44	**25.35**
	21	1	1328.43	**27.22**	1996.66	**30.39**
autogen	4	1	676.89	**657.07**	**179.27**	809.45
		2	**300.14**	968.09	994.24	**840.69**
		3	**415.00**	741.83	**756.72**	988.33
		4	**228.62**	5638.46	**800.11**	5457.23
	16	1	**2231.98**	5188.65	**420.75**	5318.42
		2	**379.31**	TO	**586.94**	TO
		3	**411.57**	6300.83	**274.74**	5459.99
		4	**771.76**	TO	**574.25**	TO
	20	1	**482.92**	4741.88	**522.16**	5573.37
		2	**548.96**	6016.29	**943.53**	6043.12
		3	**426.54**	5728.60	**945.79**	5768.85
		4	**1134.01**	TO	**568.33**	TO
cassini	5	1	**31.85**	TO	**51.11**	TO
	10	1	**82.48**	TO	**60.35**	TO
	15	1	**90.62**	TO	**71.00**	TO
	20	1	**41.50**	TO	**61.95**	TO
	25	1	**62.39**	TO	**64.06**	TO
	58	1	**58.36**	TO	**64.65**	TO
guidance	5	1	425.76	**349.59**	**196.27**	370.38
	10	1	**173.75**	1663.83	**245.50**	1727.08
	15	1	**250.78**	4616.18	**128.19**	4678.15
	20	1	**271.89**	2928.52	**300.66**	3598.55
	25	1	**224.58**	TO	**507.58**	TO
	62	1	**278.82**	TO	**95.00**	TO
pdist	5	1	**95.19**	458.85	**96.81**	350.6
		2	**46.33**	511.33	**46.72**	435.57
		3	**48.09**	424.51	**40.19**	419.86
		4	**80.44**	420.94	**29.07**	388.84
	20	1	36.72	**32.96**	1635.52	**35.92**
		2	**22.47**	28.86	35.54	**34.19**
		3	71.29	**34.42**	**34.19**	35.74
		4	33.86	**33.65**	**28.98**	31.97
	25	1	773.29	**246.48**	285.85	**280.56**
		2	**54.20**	215.76	**38.83**	279.42
		3	**35.06**	216.55	**25.86**	219.13
		4	**24.75**	217.05	**16.75**	217.25
	41	1	**23.82**	818.28	**38.74**	859.25
		2	**31.33**	643.03	**50.58**	759.50
		3	**41.38**	633.24	**14.40**	782.73
		4	**22.21**	750.93	**18.93**	818.21

References

1. Allen Emerson, E., Lei, C.L.: Temporal reasoning under generalized fairness constraints. In: Monien, B., Vidal-Naquet, G. (eds.) STACS 86. pp. 21–36. Springer Berlin Heidelberg, Berlin, Heidelberg (1986)
2. BARTHE, G., D'ARGENIO, P.R., REZK, T.: Secure information flow by self-composition. Mathematical Structures in Computer Science **21**(6), 1207–1252 (2011). https://doi.org/10.1017/S0960129511000193
3. Biere, A., Artho, C., Schuppan, V.: Liveness checking as safety checking. Electronic Notes in Theoretical Computer Science **66**(2), 160–177 (2002). https://doi.org/https://doi.org/10.1016/S1571-0661(04)80410-9, https://www.sciencedirect.com/science/article/pii/S1571066104804109, fMICS'02, 7th International ERCIM Workshop in Formal Methods for Industrial Critical Systems (ICALP 2002 Satellite Workshop)
4. Bittner, B., Bozzano, M., Cavada, R., Cimatti, A., Gario, M., Griggio, A., Mattarei, C., Micheli, A., Zampedri, G.: The xSAP Safety Analysis Platform. In: TACAS. Lecture Notes in Computer Science, vol. 9636, pp. 533–539. Springer (2016)
5. Bozzano, M., Cimatti, A., Gario, M., Tonetta, S.: Formal Design of Asynchronous Fault Detection and Identification Components using Temporal Epistemic Logic. Logical Methods in Computer Science **11**(4), (2015). https://doi.org/10.2168/LMCS-11(4:4)2015, https://doi.org/10.2168/LMCS-11(4:4)2015
6. Bradley, A.R.: SAT-Based Model Checking without Unrolling. In: VMCAI. Lecture Notes in Computer Science, vol. 6538, pp. 70–87. Springer (2011)
7. Bryant, R.E.: Binary Decision Diagrams. In: Handbook of Model Checking, pp. 191–217. Springer (2018)
8. Cavada, R., Cimatti, A., Dorigatti, M., Griggio, A., Mariotti, A., Micheli, A., Mover, S., Roveri, M., Tonetta, S.: The nuXmv Symbolic Model Checker. In: CAV. Lecture Notes in Computer Science, vol. 8559, pp. 334–342. Springer (2014)
9. Cimatti, A., Griggio, A., Magnago, E., Roveri, M., Tonetta, S.: Extending nuXmv with Timed Transition Systems and Timed Temporal Properties. In: CAV (1). Lecture Notes in Computer Science, vol. 11561, pp. 376–386. Springer (2019)
10. Clarke, E.M., Grumberg, O., Hamaguchi, K.: Another Look at LTL Model Checking. Formal Methods in System Design **10**(1), 47–71 (1997)
11. Clarke, E.M., Grumberg, O., Peled, D.A.: Model checking. MIT Press (2001)
12. Emerson, E.: Temporal and Modal Logic. Handbook of theoretical computer science **2**, 995–1072 (1990)
13. Finkbeiner, B., Hahn, C., Torfah, H.: Model checking quantitative hyperproperties. In: Chockler, H., Weissenbacher, G. (eds.) Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10981, pp. 144–163. Springer (2018). https://doi.org/10.1007/978-3-319-96145-3_8, https://doi.org/10.1007/978-3-319-96145-3_8
14. Grastien, A.: Symbolic testing of diagnosability. In: International Workshop on Principles of Diagnosis (DX). pp. 131–138 (2009)
15. Jiang, S., Huang, Z., Chandra, V., Kumar, R.: A Polynomial-time Algorithm for Diagnosability of Discrete Event Systems. IEEE Transactions on Automatic Control **46**(8), 1318–1321 (2001)

16. M. Bozzano and A. Cimatti and S. Tonetta: Testing Diagnosability of Fair Discrete-Event Systems. In: Proc. International Workshop on Principles of Diagnosis (DX-19) (2019)
17. Sampath, M., Sengupta, R., Lafortune, S., Sinnamohideen, K., Teneketzis, D.: Diagnosability of Discrete-event Systems. IEEE Transactions on Automatic Control **40**(9), 1555–1575 (1995)
18. Sheeran, M., Singh, S., Stålmarck, G.: Checking Safety Properties Using Induction and a SAT-Solver. In: FMCAD. Lecture Notes in Computer Science, vol. 1954, pp. 108–125. Springer (2000)